Shape recognition and twenty questions

Donald Geman, Bruno Jedynak

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Shape Recognition and Twenty Questions

Donald Geman, Bruno Jedynak

N° 2155

Novembre 1993

PROGRAMME 4

Robotique,

image

et vision

ISSN 0249-6399

INRIA
ROCQUENCOURT

Shape Recognition and Twenty Questions

Donald Geman, Bruno Jedynak

Programme 4 — Robotique, image et vision
Projet Syntim

Rapport de recherche n ° 2155 — Novembre 1993 — 31 pages

Abstract: We formulate shape recognition as a coding problem. There is a finite list of possible "hypotheses" - shape classes and/or spatial positionings - and we wish to determine which one is true based on the results of various "tests," which are local image features. We use a decision tree: each interior node is assigned one of the tests and each terminal node is assigned one of the hypotheses. The assignment of tests, or "strategy," is recursive: along each branch choose the next test to remove as much uncertainty as possible (as measured by entropy) about the true hypothesis. In contrast to the standard approach of "hypothesize and test," there is no repeated elicitation of hypotheses; instead, the "indexing" is dynamic and stochastic. We gradually formulate specific conjectures as the evolving distribution on hypotheses becomes increasingly peaked. We apply this "twenty questions" approach to the recognition of two types of linear, deformable structures: handwritten numerals and roads in satellite images.

Key-words: shape recognition, decision tree, entropy strategy, caracter recognition, satellite images, road-tracking

(Résumé : tsvp)

This work was partially supported by the National Science Foundation under grant DMS-8813699, by the Office of Naval Research under contract N00014-91-J-1021, and by CNES under Marché 833/CNES/93/0972/00

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

Reconnaissance de formes et jeu des 20 questions

Résumé : Nous formulons le problème de la reconnaissance de formes comme un problème de codage. D'une liste finie d"'hypothèses"- formes et/ou positionnement - nous souhaitons déterminer celle qui est réalisée (l'hypothèse vraie) à l'aide des résultats de tests (ou questions) portant sur des caractéristiques locales de l'image. Nous utilisons un arbre de décision. A chaque nœud intèrieur est assigné un test et à chaque nœud terminal est assigné une hypothèse. La "stratégie" utilisée pour assigner les tests est récursive : le long de chaque branche, choisir le test suivant de manière à réduire le plus possible l'incertitude (mesurée par l'entropie) sur l'hypothèse vraie. En contraste avec l'approche standart "hypothèse et vérification", il n'y a pas d"'essais" successifs d'hypothèses. Au contraire, la méthode pour indexer est dynamique et stochastique. Graduellement, nous formulons des "conjectures" de telle sorte que la distribution à postèriori sur l'ensemble des hypothèses devienne de plus en plus piquée. Nous appliquons cette approche à la reconnaissance de deux types de structures linéaires et déformables : les chiffres manuscrits et les réseaux routiers dans les images de satellites d'observation de la terre.

Mots-clé : reconnaissance de formes, arbre de décision, stratégie basée sur l'entropie, reconnaissance de caractères, images satellitaires, suivi de routes

1 Introduction

We explore the possibility of recognizing shapes "simply" be asking the right questions in the right order. The approach is formulated in abstract terms using statistics and information theory. We are given a finite list of possible "hypotheses" (or "states of nature"); exactly one of these is true and we wish to decide which it is based on the results of various "tests" or "questions." There is a decision tree which instructs us how to perform the tests and eventually classify the results. Each interior node of the tree is assigned one of the tests and each terminal node is assigned one of the hypotheses. The assignment of tests, the "strategy," is adaptive in the sense that the choice of the test at each node may depend on the test values observed at all preceding nodes. The strategy is regarded as a code for efficient classification. Ideally, the choice would be driven by some global measure of efficiency, such as achieving the most accurate classifier for a given average number of tests, or reaching the fastest decision at a given level of accuracy. But these problems are intractable, and we shall opt instead for the "greedy" strategy in which the tests are chosen recursively in order to remove as much uncertainty as possible about the true hypothesis.

We have applied this to two problems in shape recognition. The raw data is a single, grey-level image, the tests are particular "features" (i.e., image functionals), and the hypotheses refer to particular shape classes or spatial positionings. Specifically, we shall focus on linear, deformable structures. The main application is the tracking of major highways in SPOT satellite imagery. There is only one shape class ("road") and the hypotheses are indexed by the "pose." The strategy is constructed *on-line* based on entropy reduction. In order to illustrate the scope of the method, we shall also mention another application - the recognition of handwritten numerals; a full account will be given elsewhere. In that case, there are ten hypotheses (the pose being of secondary importance) and the strategy is constructed by the same principles, but *off-line*. Some conclusions and speculative remarks are given in the final section.

2 Twenty Questions

There is a certain parlor game which sometimes goes by the name of "Twenty Questions" and seems to be known everywhere (e.g., as "Bar-Kokheba" in Arabic or "jeu des métiers" in French). It is played like this: one player chooses an instance from a general category such as famous persons, natural objects, or historical events, and another player, who knows the category, tries to guess the particular choice by asking whether it belongs to various subsets ("is the person dead or alive?", etc.). In one version, an old American television show called "What's My Line?", the category was "unusual occupations" and there was a panel of questioners who were allowed at most twenty questions to guess the occupation of a guest.

The mathematical "Twenty Questions Problem" is to determine the optimal strategy for minimizing the mean number of questions that are asked. (It is assumed the answers are truthful.) The solution to this simplified game, and bounds on the mean decision time, are known from results in coding theory, which formalize the plan of "divide and conquer." (A precise definition of strategy will be given in §6.) Specifically, the *Huffman code* [14], [22], [26], provides an explicit construction for coding the symbols of an alphabet with (variable length) binary strings so as to minimize the mean number of bits that must be examined during decoding. The mean length of the optimal code is at most one plus the entropy of the "prior distribution," i.e., the one initially placed over the hypotheses. (Recall that entropy measures the uncertainty in a distribution, and is therefore minimized by a point mass and maximized by the uniform measure.) This length is optimal because the mean length of any code is at least equal to the starting entropy. Thus, for example, in the Twenty Questions game we would only need about twenty questions *on the average* if the category has 2^{20} items and each is equally likely to be chosen.

Of course this formulation is much too restrictive for practical decision-making problems. For one thing, the problem is too easy as stated because the hypotheses are too well separated if *every* subset question is allowed; in reality it may be impractical or impossible to have exactly the right question available at each instance, not to mention maintaining a data structure whose size is exponential in the number of hypotheses.

In the "Constrained Twenty Questions Problem"([6], [7], [17],[18]) the set of possible questions, i.e. subset choices, is limited. The problem is then determined by two things: an initial distribution on hypotheses and a binary matrix indicating the answers to the questions for each hypothesis, with 1 standing for "yes" and 0 for "no." That is, each row corresponds to an individual hypothesis and displays the answers for all tests, and each column corresponds to an individual test and picks out a subset of hypotheses, Naturally, we assume the rows are distinct (i.e., the set of tests uniquely determines the hypothesis) as well as the columns (i.e., no test is repeated). For relatively "small" problems, say for 100 hypotheses and 20 tests, one can in fact find the optimal strategy with dynamic programming. But the general problem is NP complete [13]. Still, as with many hard combinatorial optimization problems, considerable effort has been applied (see [7],[18] and the references therein) to finding good "suboptimal" strategies.

It is precisely these suboptimal strategies we intend to apply to shape recognition, but within a still more general framework that we shall still refer to as "Twenty Questions." In particular, one cannot assume that the answers are determined without error; given the true hypothesis, the tests are *nondeterministic* due to variability in the image formation process and within shape classes. In other words, the tests must be regarded as *random variables*. Moreover, they may not be conditionally independent given the hypotheses; this is the case, for example, in the application to numeral recognition; see §4 and §6.

3 Indexing

Twenty Questions is not "hypothesize and test," not in the sense of the popular approach to object recognition which proceeds as follows. Given some feature-based representation for the objects, two fundamental processes are iterated. One process is "indexing": the feature values derived from preliminary measurements are used to elicit a candidate set of objects and/or poses which is consistent with, or suggested by, the initial measurements. The other process is "matching" - a correspondence is sought between object and data features. In general these two processes are closely intertwined: indexing is driven by

trial matches and trial conjectures focus attention on additional features to compute, e.g., those likely to be visible or confirmatory.

In contrast to the repeated elicitation of candidate hypotheses., the indexing in Twenty Questions, such as it exists, is dynamic and stochastic. We *gradually* formulate specific conjectures. There is an evolving distribution on hypotheses which is updated at each node of the decision tree in light of the additional information derived from the most recent test. The distribution becomes increasingly peaked as information is accumulated, until one hypotheses (or one homogeneous group) comes to dominate.

4 Models

Twenty Questions is not "model-based" per se. What is needed is the joint distribution of hypotheses and tests. One way to determine this is to specify two distributions: the marginal (= "prior") distribution over hypotheses (which is usually just uniform) and the conditional (joint) distribution of the tests given each hypothesis. (If the tests are conditionally independent then it suffices to provide the marginal distribution of each test given each hypothesis.) In effect, our model for each hypothesis is the (conditional) distribution of tests. Moreover, we can think of the strategy as providing a *joint representation* for the ensemble of objects by selecting an ordered list of tests specifically dedicated to that ensemble.

It is necessary to estimate the conditional distribution of tests from data. In the road-finding application, since the tests are conditionally independent, the strategy may be *exactly* computed once the marginal test distributions are estimated; in particular, the conditional (= "posterior") distribution on hypotheses given the test results is calculated analytically. In contrast, in the numeral application, the tests are conditionally dependent and we must *estimate* the strategy using the empirical distribution for tests and hypotheses provided by a training set. This requires a large database of handwritten digits; otherwise we lack enough instances of test histories to properly estimate the residual entropy.

5 Related Work

Decision trees are ubiquitous. We shall mention some related types which appear in Combinatorial Optimization, Statistics, and, to a somewhat lesser extent, in Computer Vision.

As we have already seen, Twenty Questions exemplifies a class of problems in coding theory and combinatorial optimization. In particular, we shall focus below on a class of "greedy" strategies (for choosing and ordering tests) which are sometimes called "splitting algorithms" ([6],[7],[18]) in applications of binary search trees to fault-testing, machine diagnostics, and related problems.

In Sequential Design of Experiments ([3]) a more general framework is considered, which involves a loss function (i.e., a penalty for each possible misclassification), experimental "costs" for each test, and a resulting overall "risk." (Multi-armed Bandit ([10]) and Adaptive Control ([16]) problems are closely related.) A major difference from Twenty Questions is that, in these studies, the tests are *repeatable* and the emphasis is on *asymptotic* results for large numbers of tests ([3],[15]). In our case, the available information is inherently limited, represented by a *fixed number* of tests; no additional information is provided by performing the same test again: the answer must be the same because the tests are simply functionals of a *single* image.

The basic set-up in CART ("Classification and Regression Trees", [2]) is essentially the same as ours. In particular, the observations are random and the tests are not repeatable. Both approaches involve the construction of decision trees based on principles of information gain. The main difference is dimensionality - of the data, of the (effective) number of hypotheses, and of the pool of questions. For example, when object "poses" are taken into account, we are dealing with an essentially infinite number of hypotheses. Moreover, it is by no means obvious what are the "right" features to entertain, whereas, in CART, we are provided with a relatively small set of measurements and every test is derived by applying a threshold to one of these.

Decision trees have already been used for recognition in pattern recognition and computer vision. In particular, Swain ([24]) constructs a decision tree based on entropy reduction and object topology in order to recognize polyhedra. Our basic outlook is almost the same. One difference is that we account for noisy responses in the construction of the decision tree; another is that we

consider large-scale problems. Goad ([9]) uses search trees to match detected vs. predicted edges, and emphasizes pre-processing to reduce recognition time. Hansen and Henderson ([11]) use CAD models to generate a "strategy tree" based on features (e.g., edges and surface patches) and use it to generate hypotheses about object poses. Sossa and Horaud ([23]) explore a variation of geometric hashing based on relational graphs which capture the intrinsic topology of objects. See also [1], [12],[21], [25].

Finally, many ideas here originate in unpublished work by E. Bienenstock, S. Geman, D.E. McClure, and the first author on invariant (i.e., view-independent) recognition of rigid objects, such as vehicles and character fonts. A coarse-to-fine classification is performed at each image location; many hypotheses are simultaneously considered at the early stages, giving way in a controlled progression to increasingly specialized tests. The decision tree is constructed off-line by maximizing measures of dissimilarity among object silhouettes.

6 Mathematical Formulation

6.1 Hypotheses and Tests

We begin with a set of hypotheses, say $\mathcal{X} = \{x_1, x_2, ..., x_M\}$. In some cases, each $x \in \mathcal{X}$ refers to a specific object/pose pairing. For example, in our formulation of the road problem, there is one hypothesis for each (allowable) road configuration and its placement in the image. In other cases, the hypotheses are actually compound. For instance, each one might represent an entire class of shapes, as in numeral recognition; neither the poses nor specific variations are explicitly enumerated. (We may also wish to include an appropriate "null hypothesis," for example one representing the event that no object is present.) We assume that exactly one hypotheses is true, denoted by X, and chosen according to some initial distribution $\mu_0 = \{\mu_0(x) = P(X = x), x \in \mathcal{X}\}$.

Information is available from a discrete collection of tests $Y_1, Y_2, ..., Y_N$, which are random variables. In our applications, the tests are local functionals of the image data; specific examples are given later. Notice that there are two distinct sources of randomness: uncertainty about the true hypothesis and uncertainty in the test answers. In general, this family of tests is very large

and we may assume that, with probability one (or nearly one), the entire set of tests (essentially) determines X. In other words, given all the test results, the conditional distribution on X would be extremely peaked. The idea is to perform only a small fraction of the tests and still accurately estimate X. The tests are performed sequentially and adaptively, meaning that at each stage we may utilize the results of previous tests in order to choose the next one. Our problem is then to perform these tests in an "optimal" order relative to some criterion, for instance using as few tests as possible at a given accuracy level, or achieving the best accuracy with a given number of tests.

In the simplified Twenty Questions game, the random variables $Y_n, n = 1, ..., N$, are binary and conditionally degenerate. Hence all the relevant information is captured by the binary matrix $\mathcal{D} = \{D_{m,n}, m = 1, ..., M, n = 1, ..., N\}$ where $D_{m,n} = P(Y_n = 1|X = x_m)$. (Of course, $P(Y_n = 0|X = x_m) = 1 - D_{m,n}$.) There is no randomness in the test outcomes once X is known. In the case when all tests are available (so that $N = 2^M$), the mean length l of the Huffman code satisfies the inequalities

$$H(\mu_0) \leq l < H(\mu_0) + 1$$

where $H(\mu)$ denotes the entropy of the probability distribution μ:

$$H(\mu) = -\sum_i \mu(i) \log_2 \mu(i)$$

with the convention that $0 \log 0 = 0$. (The entropy of a random variable U is defined as $H(\mu)$ where $\mu(u) = P(U = u)$.) When only some of the subset questions are available (but they still determine X, i.e., the rows of $\mathcal{D}$ are distinct), the problem is then to determine the optimal strategy for minimizing $E(\tau)$, where τ denotes the decision time, i.e., the number of tests performed before X is known exactly. Unfortunately, this problem is NP complete.

6.2 Maximum Likelihood Classification

In principle, we can imagine doing all the tests and computing the maximum likelihood (or maximum a posteriori) estimator $\hat{X}^{ML} = \text{argmax}_{x \in \mathcal{X}} P(X = x|Y_1, ..., Y_N)$. This is obviously more accurate than any estimator based on doing *some* of the tests. However, even if it were feasible to do all the tests,

there remains the problem of determining $\hat{X}^{ML}$: there are certainly too many test histories to store all the possibilities in advance, and often no obvious way to compute the estimator on-line. Indeed, we wish to get roughly the same performance when each branch of the decision tree contains only a very small fraction of the total number of tests. In effect, we estimate the maximum likelihood estimator.

6.3 Strategies

For simplicity, assume each test Y_n assumes values in common, finite set $\mathcal{Y}$ of size L. A *strategy* is then a function $\pi = (\pi_1, ..., \pi_N)$ from observation vectors $(y_1, ..., y_N) \in \mathcal{Y}^N$ to permutations of $\{1, ..., N\}$ such that $\pi_1 \in \{1, ..., N\}$ is a constant, $\pi_2 = \pi_2(y_1), \pi_3 = \pi_3(y_1, y_2)$, etc. The number of possible strategies grows very fast with N.

The observations generated by re-ordering the tests $Y_n, n = 1, ..., N$ according to a strategy π will be denoted by $Q_1, Q_2, ..., Q_N$. Thus, $Q_1 = Y_{\pi_1}, Q_2 = Y_{\pi_2(Q_1)}$ and, in general, $Q_n = Y_{\pi_n(Q_1,...,Q_{n-1})}$. The notation is awkward, but the meaning should be clear: π_1 determines the first question asked, denoted Q_1; π_2 applied to Q_1 determines the next question Q_2; etc. Notice that $Q_1, ..., Q_N$ are not conditionally independent given X.

6.4 The Decision Tree

The decision tree then has Q_1 at the root (or level-one) node, which branches into L nodes corresponding to the possible answers $\{Q_1 = y\}, y \in \mathcal{Y}$. Each of these L level-two nodes then branches into L level three nodes corresponding to the outcomes of Q_2, and so forth. Certain nodes are declared to be "terminal nodes" and labeled by one of the hypotheses $x \in \mathcal{X}$. This may be formalized using stopping and classification rules but we shall skip the details. In general, each hypothesis x will be associated with many terminal nodes, and following such a terminal node back to the root will produce one sequence of observations for which $\hat{X} = x$, where $\hat{X}$ denotes our estimator of X. For each x, the totality of such sequences determines a region of observation space - the event $\{\hat{X} = x\}$.

Basically, we want to stop testing when one of the hypotheses, or one coherent class of hypotheses, becomes overwhelmingly likely. For each $1 \leq$

$k \leq N$ and each $(y_1, ..., y_k) \in \mathcal{Y}^k$ consider the "posterior distribution" $P(X = x|Q_1 = y_1, ..., Q_k = y_k)$. One simple stopping time is

$$\tau = \min\{1 \leq k \leq N | \max_x P(X = x|Q_1 = y_1, ..., Q_k = y_k) \geq 1 - \epsilon\}$$

if $P(X = x|Q_1 = y_1, ..., Q_k = y_k) \geq 1 - \epsilon$ for some $x \in \mathcal{X}, k = 1, ..., N$, and $\tau = N$ otherwise. The terminal nodes are those at "level τ." This is equivalent to stopping as soon as the entropy of the posterior distribution falls below a threshold.

The overall accuracy of the procedure might be measured by $P(\hat{X} = X)$ or using the entropy at the terminal nodes. The speed-accuracy trade-off is then represented by "dual" constrained optimization problems: fix the mean decision time and search for the most accurate decision tree (the true analogue of the twenty questions problem) or fix the level of accuracy and search for the decision tree with the shortest mean path length. In either case, exact solutions are generally unavailable and we are led to the problem of finding good heuristics.

7 The Entropy Strategy

If B is an event, the entropy of a random variable U relative to the conditional probability measure $P(.|B)$ is denoted by $H_B(U)$, and the conditional entropy of U given another random variable V is defined as $H(U|V) = \sum_v P(V = v)H_{\{V=v\}}(U)$. It follows easily that

$$0 \leq H(U|V) \leq H(U),$$

i.e., the uncertainty about U can only be reduced upon observing V, and will remain unchanged if U and V are independent.

Let $B_k = \{Q_1 = y_1, ..., Q_k = y_k\}$ and assume $P(B_k) > 0$. The basic idea is this: at stage $k + 1$ choose the test $Y_n, n \neq \pi_1, \pi_2, ..., \pi_k$, which maximizes the expected gain in information about X from observing Y_n, having already observed B_k. This is the same as minimizing the expected amount of uncertainty about X, i.e., under the pending posterior distribution. Thus,

$$\pi_1 = \text{argmin}_{1 \leq n \leq N} \; H(X|Y_n)$$

and, for $k \geq 1$,

$$\pi_{k+1}(y_1, ..., y_k) = \text{argmin}_{n \neq \pi_1, ..., \pi_k} H_{B_k}(X|Y_n).$$

A Special Case. There is a special case (that we shall apply to finding roads) in which it is possible to calculate the entropy strategy exactly. Go back to the constrained twenty questions problem and recall that if hypothesis x_m is true then the answer to test n *must* be "yes" if $D_{m,n} = 1$ and the answer *must* be "no" if $D_{m,n} = 0$. There is no randomness in the responses once the hypothesis is fixed. Now generalize this by declaring that for any pair (m, n) for which $D_{m,n} = 1$ the probability law $P(Y_n = y|X = x_m), y \in \mathcal{Y}$ for test Y_n is given by some (discrete) distribution $p_1(y)$ and whenever $D_{m,n} = 0$ the distribution is $p_0(y)$. The ideal, noiseless, answer is $Y_n = 1$ when $D_{m,n} = 1$ but the actual answer is a sample from p_1, and similarly for the other case.

Example. Fix two error rates ϵ_1 and ϵ_0 and choose $p_1(y) = 1-\epsilon_1$ and $p_0(y) = \epsilon_0$ if $y = 1$, and $p_1(y) = \epsilon_1$ and $p_0(y) = 1-\epsilon_0$ if $y = 0$. Thus ϵ_1 (resp. ϵ_0) represents the probability of hearing the "wrong" answer if the "right" answer is $Y_n = 1$ (resp. $Y_n = 0$).

In §9 we shall see that if the tests are conditionally independent, then the only quantity that matters in constructing the entropy strategy is the probability that, given the previous k answers, question Y_n is answered according to p_1.

8 Handwritten Numeral Recognition

This problem has many variations. The most realistic and difficult is to "read" unsegmented character strings in grey level images, say images of zip codes or hand-drawn checks, in which case the individual characters may touch each other and there may be artifacts from other structures. It is then necessary to deal with *multiple levels of context*. The numeral problem gets easier if one assumes (as we do) that the characters are segmented from the background and from each other.

There are now ten (compound) hypotheses corresponding to the ten digits; all variations within shape classes and spatial positionings are aggregated into

one hypothesis. Thus, $X \in \{0, ..., 9\}$ and entropy reduction is relative to X rather than an enumeration of individual presentations.

8.1 Tests

The tests are all binary and based on attributed graphs. We fix a set of possible *vertex labels*, a set of possible *relational labels*, and consider the corresponding set $\mathcal{G}$ of attributed graphs g in the usual sense. The vertex labels are invariant classes of elementary binary patterns; invariance is with respect to translation and (discrete) rotation. Some of these patterns loosely correspond to configurations resembling "endings," "junctions," and "turns," but there is no inherent subclassification problem - the tests are well-defined image functionals. For each two vertex labels, there is a set of possible invariant relations, based on properties such as coincidence (i.e., same class, same orientation), co-linearity, proximity (i.e., "next to"), reflection, etc. (In this sense, we have adopted the outlook of Lowe [19] about invariant groupings.) There are two types of tests: those asking about the existence of a specific invariant class and those asking about a relationship between two classes.

We limit ourselves to N distinct graphs. Let Y_g denote the question associated with $g \in \mathcal{G}$. Let I denote an image. There is a mechanism for determining whether the answer to the test "*Does g appear in I?*" is yes or no. For example, given a mapping from images to graphs, we put $Y_g = 1$ if g is a subgraph of $G(I)$ and $Y_g = 0$ otherwise, where $G(I)$ is the graph associated with I. Notice that the tests are conditionally dependent; for example, if g is a subgraph of f, then $X_g = 0$ implies $X_f = 0$.

8.2 Decision Tree

Let $I_1, ..., I_M$ denote the images in the database; perhaps there are several thousand of these for each of the ten digits. The conditional distribution $P(Y_g = y_g, g \in \mathcal{G}|X = x), y_g \in \{0, 1\}$, on the family of questions is estimated by the *empirical distribution* determined by the database, i.e., the proportion of images of class x which satisfy the constraints. Just as before, at each iteration we search for the test which gives the greatest average reduction in the entropy of X.

The decision tree is binary. Each node t is assigned a graph $g(t) \in \mathcal{G}$ and the two branches departing from t correspond to the two possible answers to the test Y_g associated with g. Determining the answer to subgraph questions requires considerable book-keeping, and would be computationally infeasible on-line (and perhaps even off-line) except that we restrict *a priori* the order in which the questions can be asked. For any node t, the graph $g(t)$ assigned to t must be a supergraph of the one assigned to the parent of t. In fact, we only consider strategies for which $g(t)$ differs from any ancestor graph $g(s)$ by a *single element*, meaning that either i) $g(t)$ and $g(s)$ have the same vertices and $g(t)$ has one additional relation; or ii) $g(t)$ and $g(s)$ have exactly the same relations and $g(t)$ has one additional vertex.

So the first question is the vertex label with the highest information content. Let $Q_1 = y_1, y_1 \in \{0, 1\}$. If $y_1 = 0$, we again choose from among singleton graphs (and necessarily from among the remaining vertex labels). If $y_1 = 1$, we choose $\pi_2(1)$ from among graphs g with two vertices and no relations. (Of course the two vertices may have the same label.) Continuing, if $Q_1 = 1$ and $Q_2 = 0$, then we search for $\pi_3(1, 0)$ among graphs with exactly two vertices, whereas, if $Q_1 = 1$ and $Q_2 = 1$, the next graph $\pi_3(1, 1)$ has either three vertices (one of which is "new") and no relations, or two ("old") vertices and one "new" relation. And so forth. As it turns out, most of the questions chosen are *relational*. Finally, we can store the tree since it is binary and it is possible to reach "homogeneous" nodes (i.e., one dominant explanation) with relatively few tests.

8.3 Execution

The training may be quite intensive, but we can contemplate a very large pool of tests. However, the on-line recognition is extremely simple: we need only execute the strategy, i.e., follow the instructions. *There is no on-line optimization.* Moreover, since we traverse exactly one path of the tree (or perhaps several to account for ambiguity and insure robustness) it doesn't matter how many different tests are represented altogether: we must only perform those we encounter on the path dictated by the data.

Usually, the number of questions asked varies between five and twenty. If a random strategy is employed in the construction of the decision tree, i.e., if

the tests are chosen in a random order subject to the nesting constraint, then the number needed for accurate classification can extend to 100 or more. (Of course we couldn't store a tree of that depth.) Asking *all* the questions (though not feasible) yields the maximum likelihood classifier. The trick is to get the same performance with only a tiny fraction of the tests. As mentioned earlier, a full account of the experiments will be given elsewhere.

9 Detecting Roads in SPOT Images

9.1 The Road Problem

We propose a new method for detecting major road networks in panchromatic SPOT satellite imagery with a (ground) resolution of 10 meters. The image size we consider is 1024x1024 pixels, which represents a 10km x 10km square on the ground. An example is given in Figure 3. Whereas these networks can usually be rather easily identified by people at a large scale (say on the order of several kilometers), the problem is far more difficult at small scale, as shown in Figure 3. In other words, the problem is more global than it might at first appear. (This may be one reason there doesn't seem to exist a practical solution at this time.)

Most of the algorithms previously developed contain two major components, one dealing with low-level processing and one with higher level algorithms. The low-level algorithms are devoted to the identification of local, road-like structures, often based on local filtering; see, e.g., [4]. In the second stage, local detections might be organized using geometric information (e.g., curvature) about the shape of actual roads, together with techniques from artificial intelligence, as in [5], or together with global optimization, as in [8] and [20]. In the next section we shall show how the tracking problem, that is, the problem of extracting one single road when a starting point *on* the road is given, may be solved using Twenty Questions. In addition, we are currently extending this approach to the fully automated case; this involves incorporating an appropriate "null hypothesis" to cover the case of "no road" (in the field of view) and determining appropriate rules for stopping and classification.

9.2 Tracking as a Twenty Questions Problem

A major road in a SPOT image is modeled as a discretization of a smooth, planar curve whose curvature is bounded by a known value. We use a piecewise linear approximation; each knot corresponds to a pixel and the segments (which we call "arcs") are digitized lines of approximatively the same length. Such an object is completely determined by an ordered list of regularly spaced pixels (the knots) and an algorithm which constructs a digitized line between any two pixels. The curvature constraint is expressed by limiting the angle between successive arcs. Experimental results show that if the length of the segments is sufficiently small (say 12 pixels, although this is conservative), then major roads in SPOT images can be well approximated by allowing only three angles labeled $\{0, 1, 2\}$, corresponding to "no turn" between successive segments and the next two smallest ones.

We suppose from now on that the first arc is given. Therefore each road candidate is identified with a sequence of digits in $\{0, 1, 2\}$. These are the hypotheses $\mathcal{X} = \{x_1, ..., x_M\}$, each being a road containing $\log_3 M$ arcs. The individual arcs in x are also labeled as sequences of digits; for example, if $x = 0021022011$, the constituent arcs are $0, 00, 002, 0021$ etc. Let $\mathcal{A}$ denote the set of all arcs contained in some road candidate; then $\sharp\mathcal{A} = \frac{3}{2}(M-1)$. There is a test associated with each arc $a \in \mathcal{A}$; the test values, labeled $\{0, ..., J\}$, are determined by a matched filter (see §9.5) that uses the data at the corresponding image location to measure how well the arc matches the true road. Let $\{Y_a, a \in \mathcal{A}\}$ denote this family of tests. We assume these are *conditionally independent* given X. (This is not exactly true for the functionals we use, but a reasonable approximation.) Thus,

$$P(Y_a = y_a, a \in \mathcal{A} | X = x) = \prod_{a \in \mathcal{A}} P(Y_a = y_a | X = x).$$

Another assumption is that the marginal (conditional) distributions depend only on whether or not $a \in x$, meaning that arc a is contained in road x. Let us denote these distributions by p_1 and p_0 respectively. Then

$$P(Y_a = y_a | X = x) = \begin{cases} p_1(y_a) & \text{if } a \in x \\ p_0(y_a) & \text{if } a \notin x \end{cases}$$

where $y_a \in \{0, ..., J\}$. Thus,

$$P(Y_a = y_a, a \in \mathcal{A} | X = x) = \prod_{a \in x} p_1(y_a) \prod_{a \notin x} p_0(y_a).$$

The distributions p_0 and p_1 quantify the performances of the filtering algorithm. They are estimated from sample data; see §9.5.

9.3 The Entropy Strategy

The first arc chosen is π_1, the second is π_2, which depends on the response Y_{π_1}, and so forth. The entropy strategy can now be characterized as follows. Define $\phi(z) = H(p_1)z + H(p_0)(1-z) - H(zp_1 + (1-z)p_0), 0 \leq z \leq 1$. Here $zp_0 + (1-z)p_1$ is the mixture distribution of p_0 and p_1 with weights z and $1-z$ respectively. It is easy to show that that ϕ is convex.

Theorem. *Having observed* $B_k = \{Y_{\pi_1} = y_{\pi_1}, ..., Y_{\pi_k} = y_{\pi_k}\}$ *choose the next test (i.e., arc)* $a \neq \pi_1, ..., \pi_k$ *by minimizing* $\phi(P(a \in X | B_k))$.

The proof is straightforward:

Proof: First, it is easy to see, and intuitively clear, that for any random variables U, V, and event B,

$$H_B(U, V) = H_B(U|V) + H_B(V).$$

Now take $U = X$, $V = Y_a$ and $B = B_k$, the history of the first k tests. Then

$$\begin{aligned} H_{B_k}(X|Y_a) &= H_{B_k}(X, Y_a) - H_{B_k}(Y_a) \\ &= H_{B_k}(Y_a|X) + H_{B_k}(X) - H_{B_k}(Y_a) \end{aligned}$$

Thus, the entropy strategy chooses the next test by minimizing $H_{B_k}(Y_a|X) - H_{B_k}(Y_a)$ as a function of $a \neq \pi_1, ..., \pi_k$. This value will then be $\pi_{k+1}(y_{\pi_1}, ..., y_{\pi_k})$ and the next test will be $Q_{k+1} = Y_{\pi_{k+1}}$.

$$H_{B_k}(Y_a|X) = \sum_{x \in \mathcal{X}} P(X = x | B_k) H_{B_k}(Y_a | X = x)$$

$$
\begin{aligned}
&= \sum_{x \in \mathcal{X}} P(X = x|B_k) H(Y_a|X = x) \\
&\quad \text{(since the tests} \\
&\quad \text{are conditionally independent)} \\
&= \sum_{x: a \in x} P(X = x|B_k) H(Y_a|X = x) \\
&\quad + \sum_{x: a \notin x} P(X = x|B_k) H(Y_a|X = x) \\
&= H(p_1) P(a \in X|B_k) + H(p_0) P(a \notin X|B_k)
\end{aligned}
$$

Similarly, for any $y \in \mathcal{Y}$:

$$
\begin{aligned}
P(Y_a = y|B_k) &= \sum_{x \in \mathcal{X}} P(Y_a = y|B_k, X = x) P(X = x|B_k) \\
&= \sum_{x \in \mathcal{X}} P(Y_a = y|X = x) P(X = x|B_k) \\
&= \sum_{x: a \in x} P(Y_a = y|X = x) P(X = x|B_k) \\
&\quad + \sum_{x: a \notin x} P(Y_a = y|X = x) P(X = x|B_k) \\
&= p_1(y) P(a \in X|B_k) + p_0(y) P(a \notin X|B_k)
\end{aligned}
$$

It follows that $H_{B_k}(X|Y_a) = \phi(z) + H_{B_k}(X)$ with $z = P(a \in X|B_k)$, which completes the proof.

Note: If the supports of p_0 and p_1 are disjoint, then each test result *eliminates* certain hypotheses (i.e., their posterior probabilities become zero). In this case, it is easy to see that $\phi(z) = z \log_2 z + (1 - z) \log_2 (1 - z)$, which is symmetric about $z = \frac{1}{2}$. Consequently, the strategy amounts to choosing the test a for which $|P(a \in X|B_k) - \frac{1}{2}|$ is as small as possible. In other words, we choose the arc which most nearly divides the "active" roads into two groups of equal probability. This case includes the noiseless game (e.g., constrained twenty questions) in which the value of Y_a is *determined* by X. Of course it is impossible to design a local test which discriminates perfectly between roads and background, so that the supports are never disjoint in practice.

9.4 Implementation of the Entropy Strategy

The strategy must be computed on-line since the depth of the decision tree is of order 100, corresponding to tracking approximately 10 km. Obviously the full tree cannot be computed, let alone stored. However, if we can compute it on-line, then we need only go down the path determined by the data, i.e., we needn't compute the entire tree but rather only the particular branch we traverse for the given image.

The on-line computation of the strategy is iterative. There are three steps in executing iteration $k+1$:

1. Choose an appropriate subset of arcs $\mathcal{A}_{k+1}$ from $\mathcal{A}$;

2. For each $a \in \mathcal{A}_{k+1}$, compute the quantity $z(a) = P(a \in X|B_k)$, where $B_k = \{Y_{\pi_1} = y_{\pi_1}, ..., Y_{\pi_k} = y_{\pi_k}\}$, evaluate the function ϕ at $z(a)$, and select π_{k+1} as the arc a for which $\phi(z(a))$ is minimized;

3. Perform the test $Y_{\pi_{k+1}}$, i.e., run the filtering algorithm using the image data at the location corresponding to π_{k+1}.

In the first step, $\mathcal{A}_{k+1}$ is a set of "feasible candidates" - arcs among which $\phi(z(a))$ is minimized. The first set is simply $\mathcal{A}_1 = \{0, 1, 2\}$, the three arcs branching from the (known) arc at the root of the tree. In general, it is easy to see that $\mathcal{A}_{k+1}$ need only contain the arcs that are "close enough" to $\{\pi_1, ..., \pi_k\}$. For example the set of arcs which differ only in the last two digits from one of the arcs $\{\pi_1, ..., \pi_k\}$ must contain $\arg\min_{a \in \mathcal{A}} \phi(P(a \in X|B_k))$ for reasonable choices of p_0 and p_1. In practice, $\mathcal{A}_{k+1}$ is constructed by removing π_k from $\mathcal{A}_k$ and adding a small number of new arcs. In experiments to date, the size of the set $\mathcal{A}_k$ never exceeds several hundred, whereas the total number of arcs (and hypotheses) is of order 3^{100}. Moreover, the use of pruning techniques could even further limit the growth of $\mathcal{A}_k$.

Assume now that π_k has been computed based on evaluating $P(a \in X|B_{k-1})$ for all $a \in \mathcal{A}_k$ and we wish to compute π_{k+1}. We construct $\mathcal{A}_{k+1}$ as above; now we wish to evaluate $P(a \in X|B_k)$ for $a \in \mathcal{A}_{k+1}$.

First, since the tests are conditionally independent given X,

$$P(a \in X|B_k) = (P(B_k))^{-1} \sum_{x:a \in x} P(B_{k-1}, X = x) P(Y_{\pi_k} = y_{\pi_k}|X = x).$$

Second, we can express $P(B_k)$ as a function of $P(B_{k-1})$ as follows:

$$\begin{aligned} P(B_k) &= \sum_x P(B_{k-1}, X = x) P(Y_{\pi_k} = y_{\pi_k} | X = x) \\ &= p_1(y_{\pi_k}) \sum_{x:\pi_k \in x} P(B_{k-1}, X = x) + \\ &\quad p_0(y_{\pi_k})(P(B_{k-1}) - \sum_{x:\pi_k \in x} P(B_{k-1}, X = x)) \\ &= p_0(y_{\pi_k}) P(B_{k-1})(1 + P(\pi_k \in X | B_{k-1})(v(y_{\pi_k}) - 1)) \end{aligned}$$

where

$$v(y_{\pi_k}) = \frac{p_1(y_{\pi_k})}{p_0(y_{\pi_k})}.$$

Since π_k necessarily belongs to $\mathcal{A}_k$, we can easily compute the new normalizing constant $P(B_k)$.

Finally, we exploit the fact that the arcs reside on a tree to show that, for any $a \in \mathcal{A}_k \cap \mathcal{A}_{k+1}$, we can compute $P(a \in X | B_k)$ in terms of $P(a \in X | B_{k-1})$. (This covers nearly all the arcs $a \in \mathcal{A}_{k+1}$; for the few remaining a direct computation is easy since we have the normalizing constant.) For any two arcs $a, b \in \mathcal{A}$, there are three possible cases:

1. There is no $x \in \mathcal{X}$ which contains both a and b;
2. Arc a is a descendant of arc b, i.e., $a \in x$ implies $b \in x$ for any x;
3. Arc b is a descendant of arc a.

Now take $b = \pi_k$. In case (1),

$$\sum_{x:a \in x} P(B_{k-1}, X = x) P(Y_{\pi_k} = y_{\pi_k} | X = x) = p_0(y_{\pi_k}) P(B_{k-1}) P(a \in X | B_{k-1})$$

which implies that

$$P(a \in X | B_k) = \frac{P(a \in X | B_{k-1})}{1 + P(\pi_k \in X | B_{k-1})(v(y_{\pi_k}) - 1)}$$

Similarly, in case (2),

$$P(a \in X | B_k) = \frac{v(y_{\pi_k}) P(a \in X | B_{k-1})}{1 + P(\pi_k \in X | B_{k-1})(v(y_{\pi_k}) - 1)}$$

and in case (3),

$$P(a \in X|B_k) = \frac{P(a \in X|B_{k-1}) + P(\pi_k \in X|B_{k-1})(v(y_{\pi_k}) - 1)}{1 + P(\pi_k \in X|B_{k-1})(v(y_{\pi_k}) - 1)}.$$

9.5 Tests: Local Filtering

The filtering algorithm is designed to identify short, linear segments that are likely to lie on major roads. The basic assumption that is utilized is that two pixels in close proximity and both on the road should have a smaller intensity difference than that of two pixels, one of which lies on the road and the other off the road. Since the major roads appear in SPOT images as structures of width between two and three pixels, the filter is based on a pair of adjacent, digitized lines (see Figure 1). The overall res-

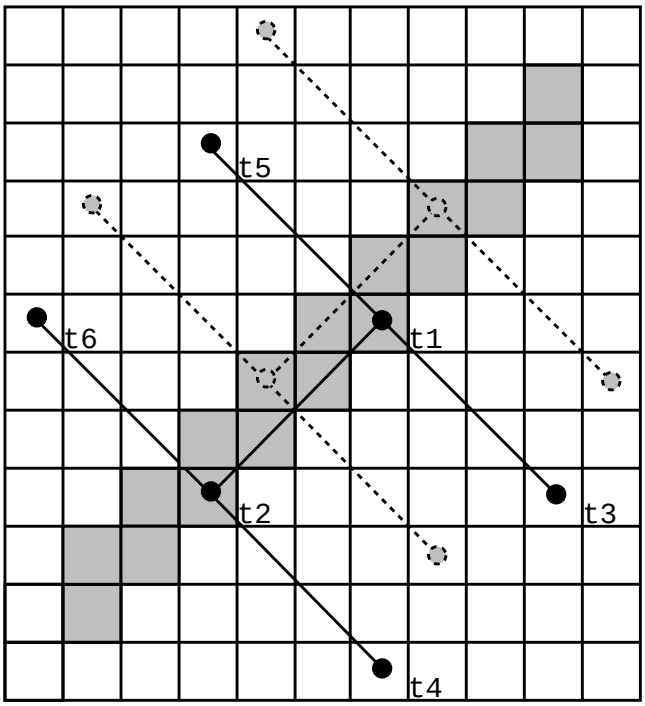

Figure 1: Local Filtering.

ponse of the filter is the aggregate along the strip of the individual, binary responses of elementary probes, each based on a pattern of six pixels $(t_1, ..., t_6)$ arranged as shown in Figure 1. Let $I(t)$ denote the image intensity at pixel t. Each such pattern is assigned a value 0 or 1; the value is 1 if $|I(t_1) - I(t_2)| < \min\{|I(t_3) - I(t_1)|, |I(t_5) - I(t_1)|, |I(t_4) - I(t_2)|, |I(t_6) - I(t_2)|\}$ and 0 otherwise. The test result is the average value along the strip, which is renormalized to obtain a value in $\{0, ..., J\}$.

The distribution of the filter output is p_1 for "true" strips and is p_0 for strips off main roads. These distributions were estimated using several test images. (The smaller roads appearing in the images were taken into account when estimating p_0.) See Figure 2.

9.6 Experiments on SPOT Images

Results are presented in Figures 4,5 and 6. The three images show three different areas in France, around Toulouse, La Rochelle, and Mantes, respectively, and are of increasing complexity. The white lines which are superimposed on the original images show the (arc) locations where the tests were performed. In Figure 4, the starting point was chosen on the North side and about 200 tests were performed during the tracking. The second image (Figure 5) is more difficult; the main road that goes from East to West is not as visible as the one in the previous case (see Figure 3). Morever the road crosses a town and becomes essentially invisible for about 200 meters (20 pixels). The starting point is chosen on the West side. Tracking gets "stuck" in the Northeast corner, and would need to be re-initialized to continue. The third image (Figure 6) is quite difficult: the main road itself is somewhat hard to see and there are many competing structures such as small roads and rivers. The starting point was chosen on the East side. In all three cases, the tracking time is on the order of 10 seconds on a SUN-SPARC computer. Finally, there are no parameters to choose; in particular, the distributions p_0 and p_1 are fixed throughout.

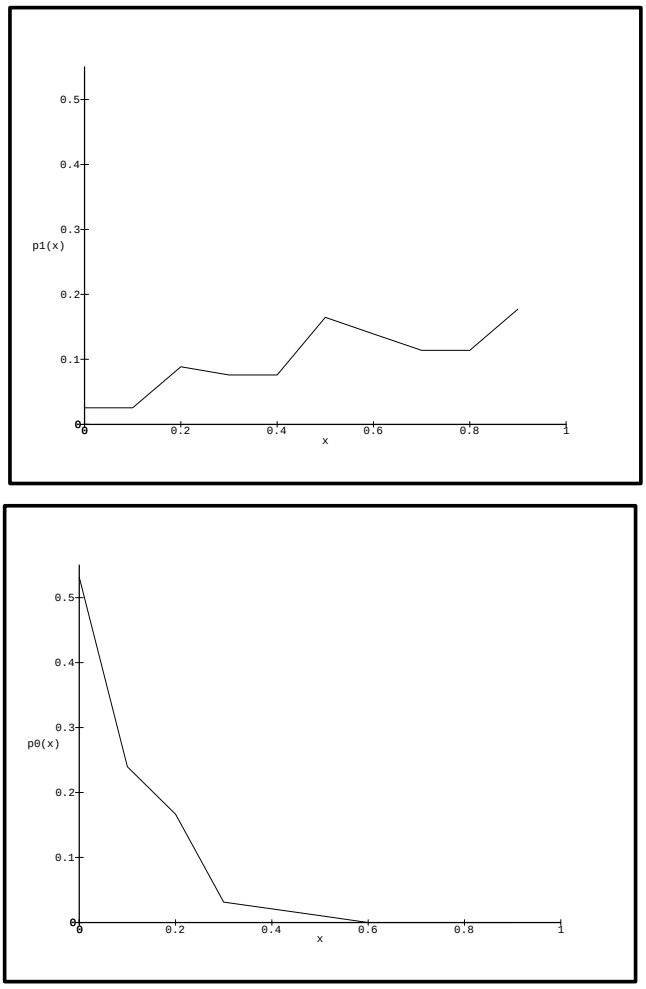

Figure 2: Estimated Distribution of the Filter. Top : On Roads; Bottom: In Background

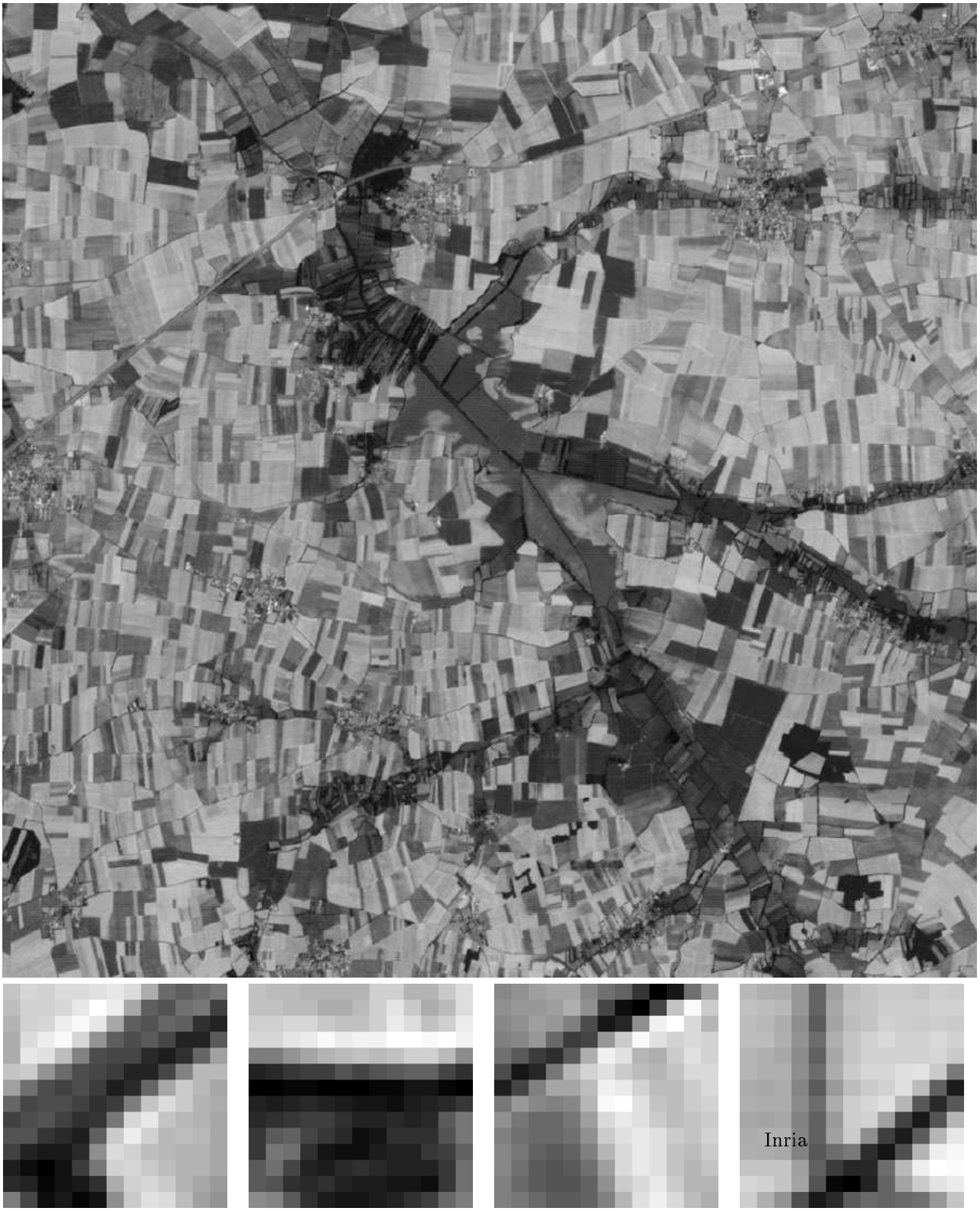

Figure 3: Top: A 1024x1024 SPOT image of La Rochelle, France. Bottom: Four subimages: the two on the left are on the main road and the two on the right are from the background.

Figure 4: Tracking results on a SPOT image from Toulouse, France. The arcs selected by the algorithm are shown in white.

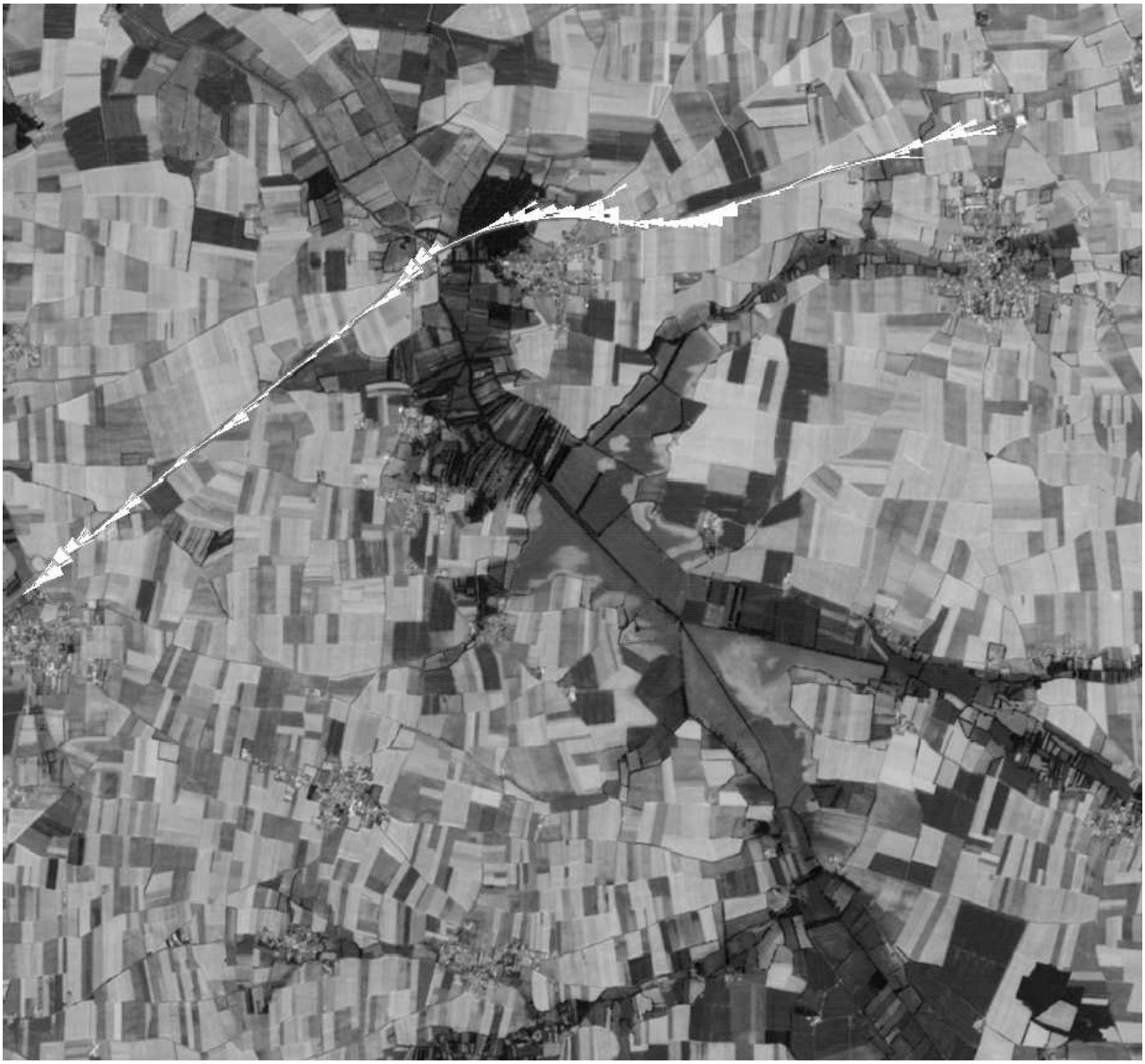

Figure 5: Tracking results on a SPOT image from La Rochelle, France. The arcs selected by the algorithm are shown in white.

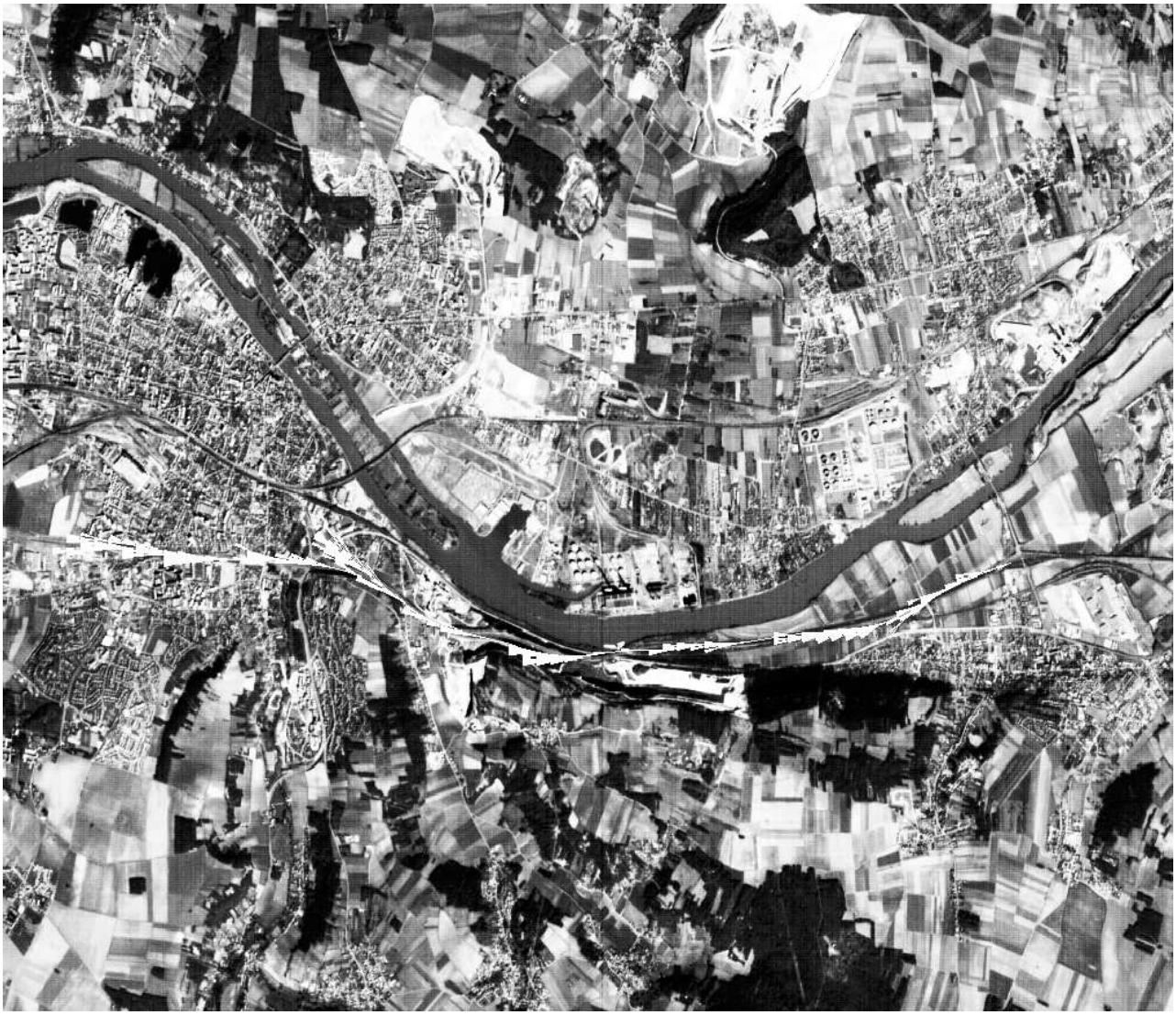

Figure 6: Tracking results on a SPOT image from Mantes, France. The arcs selected by the algorithm are shown in white.

10 Conclusions

Perhaps the main contribution here is the demonstration of how Twenty Questions (basically "divide-and-conquer") may be applied to high-dimensional recognition problems. In handwritten numeral recognition, the decision tree is computed off-line. Each test is associated with an attributed graph; the computation is manageable because we restrict the amount of "new" information in each test and because we exploit the discriminating power of *relational* questions. The resulting decision tree is then of modest depth. In contrast, the strategy for tracking roads is computed on-line, and, in fact, most of the computation is devoted to choosing, rather than executing, the tests. In this case we exploit a convenient analytic characterization of the entropy strategy and certain recursion properties.

Both applications are incomplete. For example, we assume the numerals are isolated, i.e., segmented from the background and from each other, and we assume we begin on the road, so the problem is essentially one of tracking. In order to achieve more ambitious goals, especially in the case of unconstrained script recognition, any approach must solve the fundamental problem that, at some level, the data is ambiguous and it is necessary to incorporate multiple levels of context. Such issues might stand in the way of extending Twenty Questions to very general object recognition problems.

11 Acknowledgments.

The road detection work was carried out at Project Syntim at INRIA-Rocquencourt; we are grateful to Andre Gagalowicz for making resources available and to Jean Philippe Roze for showing us how to make the on-line computations feasible. We also thank Keith Hartt for carrying out the preliminary experiments on numerals. Finally, the first author wishes to cite many discussions with E. Bienenstock, S. Geman and D.E.McClure about the pros and cons of competing recognition paradigms, and to thank S. Geman for convincing him (although not himself) that Twenty Questions was as good as any.

References

[1] Arkin, E., Meijer, H., Mitchell, J., Rappaport, D., and Skiena, S. "Decision Trees for Geometric Models," Proc. Ninth ACM Symp. on Computational Geometry, 1993.

[2] Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees* Wadsworth, Belmont, CA., 1984.

[3] Chernoff, H., *Sequential Analysis and Optimal Design* SIAM, Philadelphia, Penn. 1972.

[4] Duda, R.O. and Hart, P.E, "Pattern Classification and Scene Analysis",New York :Wiley, 1973.

[5] Fischler, M., Tenenbaum, J. and Wolf, H., "Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique," Computer Graphics and Image Processing, 15, 201-223, 1981.

[6] Garey, M.R., "Optimal binary identification procedures," SIAM J. Appl. Math., 23, 173-186, 1972.

[7] Garey, M.R. and Graham, R.L., "Performance bounds on the splitting algorithm for binary testing," Acta Informatica, 3, 347-355, 1974.

[8] Geman, D. and Jedynak, B., "Detection of roads in SPOT satellite images," Proc. IGRASS 91, Helsinski, 1991.

[9] Goad, C. "Special purpose automatic programming for three- dimensional model-based vision," Proc. Image Understanding Workshop, ARPA (1983) 94-104, 1983.

[10] Gittins, J.C., *Multi-armed Bandit Allocation Indices* John Wiley and Sons, 1989.

[11] Hansen, C. and Henderson, T., "Towards the automatic generation of recognition strategies," Second International Conf. on Computer Vision, IEEE, 275-279, 1988.

[12] Hartmann, C., Varshney, P., Mehrotra, K. and Gerberich, C., "Application of information theory to the construction of efficient decision trees," IEEE Trans. Info. Theory, 28, 565-577, 1982.

[13] Hyafil, L. and Rivest, R., "Constructing optimal binary decision trees is NP-complete," Information Processing Letters, 5, 15-17, 1976.

[14] Huffman, D.A., "A method for the construction of minimum redundancy codes," Proc. I.R.E. 40, 1098-1101, 1952.

[15] Keener, R., "Second Order Efficiency in the Sequential Design of Experiments", Ann. Stat., 12, 510-532, 1984.

[16] Kumar, P.R., "A Survey of Some Results in Stochastic Adaptive Control", SIAM J. Control and Optim., 23, 329-380, 1985.

[17] Lawler, E., *Combinatorial Optimization: Networks and Matroids*, Saunders College Publishing, 1976.

[18] Loveland, D.W., "Performance bounds for binary testing with arbitrary weights," Acta Informatica, 22, 101-114, 1985.

[19] Lowe, D.G., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Boston, 1985.

[20] Merlet, N. and Zerubia, J., "A Curvature Dependant Energy Function for Detecting Lines in Satellite Images" SCIA, Tromsø, June 93.

[21] Quinlan, J.R. and Rivest, R.L., "Inferring decision trees using minimum description length principle," Information and Computation, 80, 227-248, 1989.

[22] Sandelius, M., "On an optimal search procedure," Am. Math. Monthly, 68, 133-134, 1961.

[23] Sossa, H. and Horaud, R., "Model-indexing: the graph-hasing approach," IEEE Conf. Comp. Vision Patt. Recog., Champaign, Ill., 1992.

[24] Swain, M., "Object recognition from a large database using a decision tree," in Proc. of the DARPA Image Understanding Workshop, Morgan Kaufman, 1988.

[25] Wang, Q.R. and Suen, C.Y., "Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition," IEEE Trans. PAMI, 6, 406-417, 1984.

[26] Zimmerman, S., "An optimal search procedure," Am. Math. Monthly, 66, 690-693, 1959.

Unité de recherche Inria Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 Villers Lès Nancy
Unité de recherche Inria Rennes, Irisa, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Unité de recherche Inria Rhône-Alpes, 46 avenue Félix Viallet, 38031 Grenoble Cedex 1
Unité de recherche Inria Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Unité de recherche Inria Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex

Éditeur
Inria, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
ISSN 0249-6399